data types (e.g., aspect, query, key, action, structure, relation, service module group, and aspect group).

[0049] The data model for attributes of aspects, queries, keys, and actions is based on structure descriptors **74**. In one example, every aspect has one structure descriptor **74** that defines the data attributes of the aspect. Structure descriptors **74** refer to a data dictionary in repository **18**. A data dictionary is a collection of descriptions of the data objects or items in a data model for the benefit of programmers and others who need to refer to them. The structure descriptors **74** can be defined in an XML Schema or in one or more database tables in repository **18**.

[0050] In one example, structure descriptors **74** defined in repository **18** include flat structures in database tables. A flat structure is a sequence of pairs of attribute names and field descriptors **86** of simple value types such as real, integer, character string, and boolean. For instance, a structure descriptor **74** defining a two dimensional point can be a list {X, real, Y, real}, where X and Y are attribute names having real values.

[0051] In another example of the repository **18**, structure descriptors **74** can include nesting and collections of other structure descriptors **74**. Nesting of other structure descriptors **74**, or sub-structures, to enable the generation of larger aspects is useful whenever the use of keys for sub-structures defining smaller aspects does not make sense.

[0052] For front end application program **12** to access data (e.g., data stored in backend database **24**) from service providers **26** through the service manager **16**, instances of business object classes are identified by unique keys within a service module, for example the number of an order or the id of a product. To differentiate between different types of keys for different aspects in a service module, key descriptors **64** define different types of keys. A key descriptor **64** is associated with a structure descriptor **74** that can include more than one data attribute. In one example, every key has a character string attribute. A service module can be associated with different key descriptors **64** for different aspects, e.g., an order key may have another key descriptor **64** as an order item key.

[0053] Service module descriptor **54** includes a collection of aspect descriptors **56**. An aspect descriptor **56** refers to one structure descriptor **74** and one key descriptor **64**. The structure descriptor **74** includes all key attributes of the corresponding key descriptor **64**. Key descriptors **64** are specialized aspect descriptors **56**. The key descriptor **64** attribute of a key refers to itself as a self-reference. Examples for aspect descriptors **56** within a simple sales order service module can include: Order, Order Detail, Shipping Address, Billing Address, and Order Item as well as descriptors for key aspects like Order ID and Order Item Key. Service module descriptor **54** specifies the collection of supported aspect descriptors **56**. Multiple service module descriptors **54** can be associated with the same aspect descriptor **56**.

[0054] Aspect descriptors **56** relate to each other specified by relation descriptors **84**. A relation descriptor **84** has one source aspect descriptor **56** and one target aspect descriptor **56**. In this sense, relation descriptors **84** are directed. Relation descriptors **84** also have an optional cardinality (e.g., 1 . . . n) and a category. Supported categories are, for example, Parent-Child or Child-Parent.

[0055] A relation descriptor **84** defining a relation between source aspect A and target aspect B means that it is possible to traverse from instances of aspect A to instances of aspect B. For example, given that aspects A and B are implemented in backend database **24** as relational database tables, this means that one or more fields in a table corresponding to aspect A point to the primary key of a table corresponding to aspect B. The relation descriptor **84** defining a Parent-Child relation from source aspect A and target aspect B means that aspect B depends on the existence of aspect A. For example, given that aspects A and B are implemented in backend database **24** as relational database tables, this means that a primary key of a table corresponding to aspect B is derived from a table corresponding to aspect A. Relation descriptors **84** are introduced to describe internal navigation from one aspect to another within the same service module, e.g., from an order to the shipping address (cardinality 1 . . . 1) or to the order items (cardinality 1 . . . n) within a sales order service module. Relation descriptors **84** are independent of service modules and can be reused by different service modules. For an internal navigation or traversal from one data type to another in backend database **24**, the visible (usable) relation descriptors of a source aspect descriptor **56** are defined by the service module descriptor **54**, which has a list of supported relation descriptors **84**. Navigation is allowed for those supported relation descriptors **84** that have a target aspect descriptor **56** that is also supported by the service module descriptor **54**.

[0056] Operations for accessing and acting on data types in backend database **24** are described in operation descriptors **70**. The structure descriptor **74** defines input parameters of the operation descriptor **70**. This structure descriptor **70** also includes an input key descriptor **64** that enables mass and filter operations. Mass operations are operations specified by front end application program **12** on multiple instances of a data type in backend database **24**. Filter operations filter the results of an operations, e.g., a query, by the keys defined by the input key descriptor. Input parameters for operation descriptors **70** are optional.

[0057] There are three types of operation descriptors **70** i.e., query descriptors **104**, aspect action descriptors **92**, and action descriptors **96**. The aforementioned commands Query and Execute action are defined by operation descriptors **70**.

[0058] Query descriptors **104** describe query methods that allow searching for instances of aspects within a service module. The query descriptor **104** includes an input parameter, an input key descriptor **64**, and a result aspect descriptor **56**. The input parameter is a structure descriptor **74** that defines the search parameter structure of the query. The input key descriptor **64** defines which keys may be used for filtering. For example, executing a query defined by a query descriptor **104** with filtering keys results in a list of keys meeting the criteria of the first input. This list of keys is filtered by the set of filtering keys of the input key descriptor **64** so that a subset of the list of keys can be returned. The result aspect descriptor **56** for the query descriptor **104** specifies the type of result of the query, which could be any aspect descriptor **56** that is associated with the service module.

[0059] Each service module descriptor **54** has a set of supported query descriptors **104**. Service module descriptors **54** cannot use query descriptors **104** defined in other service